Research Inventy: International Journal Of Engineering And Science
Vol.05, Issue 01 (January 2015), PP: 09-14
Issn (e): 2278-4721, Issn (p):2319-6483, www.researchinventy.com

Reduce the False Positive and False Negative from Real Traffic with Intrusion Detection

Pon. Arivanantham, Dr. M.Ramakrishnan
Research Scholar, Sathyabama University, Professor,Velammal Engineering College

ABSTRACT *- In a typical network, the traffic through the network is heterogeneous and consists of flows from multiple applications and utilities. Considering today threats in network there is yet not a single solution to solve all the issues because the traditional methods of port-based and payload-based with machine learning algorithm suffers from dynamic ports and encrypted application. Many international network equipment manufactures like cisco, juniper also working to reduce these issues in the hardware side. Here this paper presents a new approach considering the idea of service-based. This method is, in some sense, orthogonal to current approaches and it can be used as an efficient complement to existing methods to reduce computation and memory requirements. Experimental results on real traffic confirm that this method is extremely effective and may improve considerably the accuracy of traffic classification, while it is suitable to a large number of applications. Finally, it is also possible to adopt a service database built offline, possibly provided by a third party and modeled after the signature database of antivirus programs, which in term reduce the work of training procedure and over fitting of parameters in case of parametric classifier of supervised traffic classification.*

Index terms *– network operations, security, traffic classification.*

I. INTRODUCTION

Traffic classification is one of the hottest topics in computer networks. On the one side, network managers want to know precisely the type of traffic transmitted over their networks to enforce various polices such as for quality of service (QoS), security, management, and more. On the other side, an increasing number of applications tend to hide their behavior (through encryption, tunneling, etc.) trying to avoid limitations imposed by such policies. Many international network equipment manufactures like cisco, juniper also working to reduce these issues in the hardware side Traditionally, traffic classification relies on the port based method, which exploits transport layer information (source and destination TCP/UDP ports). However, this method has many limitations that make it quite imprecise and inefficient despite its extensive usage. Not all servers respect well-known ports conventions, malicious software can use well-known ports in order to let its traffic pass through port-based security restrictions, many peer-to-peer applications actively try to avoid classification using random ports, network tunnels can be instantiated using well known ports in order to avoid imposed traffic restrictions, IP payload encryption hides the port numbers. An evolution of this approach relies on payload-based inspection that is used in most commercial devices and is declined in different flavors [4]. This technique shares some of the problems of port-based classification (encrypted protocols, tunneling) and is perceived as really expensive from the computational point of view. Other classification techniques that aim at identifying applications based on their behavior as inferred from observed traffic (statistic traffic analysis or heuristic analysis) are being studied, but are far from being ready for commercial deployment. This paper presents a new classification technique that, in some respect, is orthogonal to the above mentioned mechanisms. Our approach, called *service-base classification*, exploits information about services previously discovered in the network in order to classify traffic flows. Main advantages of this method are robustness, accuracy, a limited use of processing power, reduced memory requirements, and the capability to use any classifier in the early stage of the classification (namely, the *service identification* phase).

II.RELATED WORK

Currently deployed network classification algorithms generally fall in one of two categories: payload based algorithms and behavioral algorithms. This section provides a brief overview of the state of the art in network traffic classification focusing on some of the most relevant algorithms in each category.
Payload-based classification is applied by most commercial solutions for various purposes ranging from statistics to security, because it provides the best trade-off between the classification accuracy and the coverage in terms of number of recognizable protocols. A possibly deep inspection of data transported within packets is used to identify the flow packets belonging to and the application generating it. In fact, by inspecting the headers of the higher layer protocols, possibly up to the application layer payload, it is

possible to precisely identify the protocol being used by the application possibly gather information on the type of traffic it generates. However, the correct identification of a protocol is not straightforward. One approach relies on searching for patterns or regular expressions that can uniquely identify each protocol; a database containing the description of each protocol is needed. Many payload based solutions have been proposed [2] [3], some coupled with an approach for describing network protocols in order to make classification code easy to reuse and update [5][6]: classification of additional protocols or new versions of existing protocols can be achieved by simply adding their description, without the necessity of any modification to the classification software itself. Known problems of payload based classification algorithms are (*i*) high sensitivity to packet loss and TCP/IP fragmentation and segmentation issues, (*ii*) hard and time-consuming task of creating protocol signatures, that are crucial to the effectiveness of the solution, (*iii*) encryption and/or tunneling that hinders access to data contained into application layer headers and payloads, and (*iv*) significant requirements in terms of computational and memory resources that actually make traffic classification at high line rates difficult. Due to the high computational requirements of deep packet inspection, payload based classification algorithms usually limit pattern searching to the initial packets of each flow. According to this method, named Packet Based–Flow State in [4], once the protocol transported by a flow has been recognized, the flow identifier (i.e., the 5-tuple including IP addresses, ports, and transport layer protocol) and the corresponding application-layer protocol are added to a data structure in memory, often called *session table*, that is maintained as long as the flow is active[1].The main critic moved toward these methods is about the memory usage for maintaining flow state information; in case of large networks, the size of such per-flow state grows significantly and this might become an issue. Furthermore, additional memory is required because pattern matching usually relies on regular expressions, which are well-known for their memory consumption due to the necessity of maintaining graph-based structures representing Deterministic Finite Automata.

III. SERVICE-BASED CLASSIFICATION

Service-based classification is a surprisingly simple idea that relies on the observation of how hosts usually interact and on the assumption that certain hosts, usually called *servers*, perform similar interactions, usually offering a *service*, with multiple other hosts over a certain time span. This assumption, which provides the foundation of our method, will be verified through experiments on real network data in Section V.B. According to the classic client-server paradigm, a potentially large number of hosts connect to a single one to obtain a service. In this situation it is easy to identify the server as a main actor with a long lasting role as it usually offers the same service at the same "network coordinates" (IP address and TCP/UDP port) for a long time. The basic assumption in service-based classification is that knowing which service is offered at an IP address/port pair, a classifier can infer that all sessions directed toward that pair will access such service. For example, if the classifier knows that host www.polito.itis running a web server on TCP port 80, it can classify all sessions established to this IP address/port pair as HTTP traffic. It is important to notice that such a classifier does not work like a port based classifier. While the latter assumes that a session is transporting HTTP because it is connected to TCP port 80, a service-based classifier *knows* that www.polito.itis running a web server on TCP port 80.When the classifier discovers a service, it stores the triple identifying it — i.e., IP address (of the server), TCP/UDP port (at the server), and transport protocol in an appropriate structure in memory called *Service Table*. Service-based classification features interesting advantages over other classification methods. *Encrypted traffic* at application layer can be properly classified provided that the corresponding service has been previously identified, i.e., it has an entry in the service table. It offers *pattern segmentation transparency*, i.e., a flow can be properly classified even though protocol identifying patterns are split across multiple packets, avoiding the complexity of reassembling application data units. A service-based classifier needs to maintain only information about services (i.e., IP address, port, transport protocol and service offered) independently of the number of traffic flows actually using such services; hence it has *limited memory requirements*. The limited amount of state information kept bya service-based classifier impacts (*i*) *scalability*, performance in terms of (*ii*) *lookup time* and (*iii*) *hardware implementations* that can rely on faster on-chip memory. Service-based classification also has some potentially critical issues. Its effectiveness, in terms of minimizing both misses and wrong matches, and also its performance heavily depends on identification of network services that must be as accurate as possible.

IV. IMPLEMENTING A SERVICE-BASED CLASSIFIER

Although the service-based classifier looks simple and elegant, some issues need to be addressed to make it working properly. This section presents such issues and gives some insight in how they have been addressedin our implementation. Given the generality of the service-based method, other implementation strategies can be adopted.

A. Service identification

Given the expertise and previous work of the authors, a payload-based implementation of a service identification module has been an obvious choice. In particular, an existing packet processing engine based on the Network Packet Description Language (*NetPDL*) [1] [5] has been reused in the implementation of the service identification module. NetPDL is an application-independent packet format description language that enables the creation of a generic protocol description database: the NetPDL database, in fact. Although it includes only packet header formats and does not support the description of protocol temporal behavior (e.g., a protocol state machine), it has proved being extremely effective and robust with respect to traffic classification [4], thanks to an extension that enables management of lookup tables, originally used to maintain transport-level sessions [5]. The high flexibility of NetPDL makes the engine suitable for the implementation of the service-based classifier as well, in addition to the payload-based service identification module. The main modification made to the NetPDL engine is the addition of some new tables, such as the service table that contains information about services. The process starts with an empty service table, while traffic is processed by extracting IP addresses and ports from each arriving packet.

B. Distinguishing clients and servers

The server side of a TCP session can be easily identified by observing the SYN and ACK flags during in the three-way handshake of the TCP protocol. In our implementation we use an additional lookup table, called *Candidate Service Table*, in which a new entry is added with the IP address and port of a host that accepted an unclassified TCP session by generating a TCP packet with both the SYN and ACK flag enabled. The Candidate Service Table is required to keep track of the server side of a session because the service is possibly identified, e.g., through payload inspection, once the session has been opened, i.e., when the SYN/ACK flags, used only during the initial handshake phase, are not available to enable the identification of the server side. When the service is finally identified, the server information is moved from the candidate service table the service table.

Entries of the Candidate Service Table are subject to a very fast ageing (about ten seconds [19]) in order to avoid their number to explode over time due to sessions opened by unidentified services, unsuccessful handshakes, or unused opened sessions, as in cases of malicious activity such as SYN flooding and port scanning.

UDP traffic classification, that requires a non-straightforward extension of what is proposed in this work, is left to a companion future paper.

C. Managing the service table

Besides properly populating the service table, an important issue is the prompt elimination of service entries once the corresponding service is no longer provided. This is important in order to avoid the explosion of the number of service entries and that a service offered only temporarily leads to classification errors. One possible approach is to purge an entry that does not make a hit for a certain amount of time, hereafter referred to as *service inactivity timeout*. Typical examples are peer-to-peer applications. Assigning distinct service inactivity timeouts to different classes of services, although not strictly necessary, is useful in avoiding multiple re-identification of long-term services, e.g., through costly deep packet inspection. On the other hand, assigning an entry to the long-term service category is critical because if the service is not actually long-term or it has been wrongly identified, the entry can lead to persisting classification errors. Consequently, there should be a certain level of certainty about service before categorizing it as a long-term one. One possible policy is to set any newly identified service "under observation": its entry is categorized as short-term and some additional checks are performed on packets classified according to the entry.

V.EXPERIMENTAL EVALUATION

This Section provides an experimental evaluation of service-based classification, including some problems that arise in its implementation. The next section first devises the benefits expected by the deployment of service-based classification from an analysis of network traffic itself — i.e., not based on the results of particular classification experiments — which provides a more general assessment of the potential of service based classification. Then, the results of specific classification experiments are reported to substantiate such general assessment.

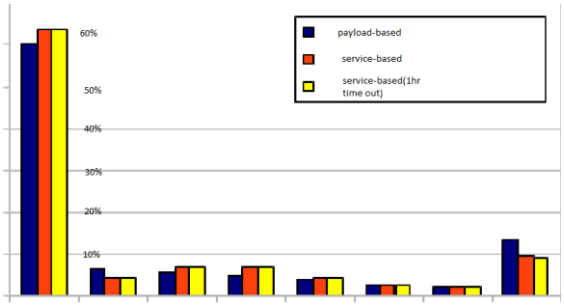
A. General Assessment

Before implementing our service-based classifier we collected a set of session-related statistics on the link that connects our University to the Internet to assess the potential benefits of service-based classification in terms of memory occupancy, i.e., if the number of services was really smaller than the

number of sessions. These measurements, done using *Tstat* [15] and lasting several days, wanted to determine the maximum number of service entries required to classify all the traffic with a service-based approach, compared to the number of session entries required by a classifier based on session identification. Figure 2 shows, for each minute, the number of active traffic sessions and the corresponding number of services on the uplink (100 Mbps) of our university network (about 6,000 hosts) over a 7-day period. The average number of active traffic sessions is 80,000 with peaks of 180,000, while the total number of services never exceeds 10,000. Figure 3 shows the same figures for a traffic trace[2] from the MAWI wide traffic archive [21]. The average number of active session is 120,000 with a peak of 380,000, while the total number of services never exceeds 10,000. The average on the whole observation period of the *session to service* ratio is about 20 for both traces, which means that a service table requires roughly 20 times fewer entries than a session table.

Observed sessions	40503
Observed services	21675
Observed applications	81
Services in which sessions are classified univocally as	21042

The tool has been installed on 11 hosts (with Linux, Windows and MacOS-X operating systems, running several applications; among the other Skype, Emule, Joost, uTorrent), the traffic produced has been captured for 4 days and the traffic traces have been analyzed by a payload-based classifier.

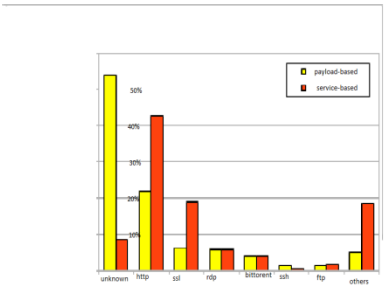


A. Comparision

An important observation is that simply increasing the service inactivity timeout may not be a good idea, since we may end up filling the service table with entries related to one- shot services or services that are anyway not any longer active, which will never appear again in the future. This is evident in Figure 7 that shows an almost four-fold increase of the service table size when changing the service inactivity timeout from 10 to 60 minutes— without any appreciable advantage in terms of classified traffic, as shown by Figure 6. Therefore, a 10 minute service inactivity timeout has been used in the experiments producing all the results presented in this discussion.

B. Accuracy

Our tests show that service-based classification offers an improvement in classification accuracy over results obtained with the original payload-based classifier. For example, trace *Weekend* contains a significant amount of traffic generated by eDonkey that hinders payload-based classification when application-layer data is encrypted. The payload-based classifier recognized only a small percentage of the flows generated by these applications, e.g., some sessions that are occasionally sent in clear and that represent special cases. For example, Skype sometimes produces packets that are only partially encrypted and consequently can be properly inspected and classified; similarly, not all eDonkey messages are encrypted. In all the other cases, the payload-based classifier is unable to identify the protocol transported and it marks flows as unknown, as it is shown by the high percentage of unknown traffic in Figure 9. Experimental evaluation also showed another problem related to the completeness of the pattern database used by the payload- based method.This is confirmed by Figure 9 where the service-based classification leaves a much smaller amount of traffic as unknown, while classifies as eDonkey a much larger percentage of traffic than payload-based classification.

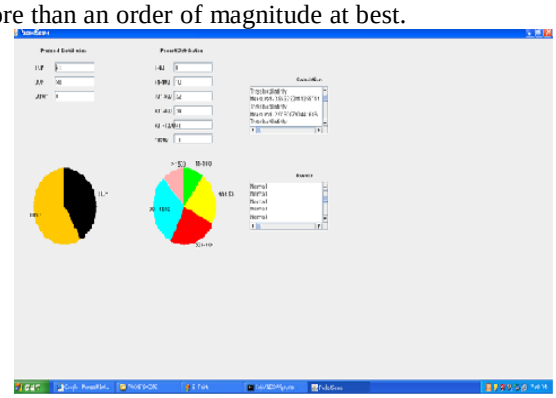


Comparison of both service and payload-based classification

C. Scalability

Scalability must be assessed in terms of memory and processing requirements.

From the processing side, the computational complexity of a classification solution is an important index of its scalability. Profiling done on our classification code (written in C/C++) confirmed that the cost for a lookup in the service table (i.e. the main cost associated to each packet by the service-based method) is 37 times lower than the cost for a pattern matching on the payload (9700 clock ticks against 260^7). Although the asymptotic processing cost remains the same in both service- based and payload-based classifier (in the unfortunate case in which each service is associated to a single session), in practical terms our method guarantees a speed-up of more than an order of magnitude at best.



Classification of traffic

VI. CONCLUSIONS

This paper presents a new idea for traffic classification, named *service-based classification*, that is, in some respect, orthogonal to the other classification techniques. This method introduces also in the traffic classification arena the concept of *fast path*, through which the vast majority of the traffic is processed with a limited use of processing and memory resources —ultimately in a short time— and a *slow path* that is invoked in a limited number of cases. In this respect, service-based classification aims at providing a solution to the fast-path processing by proposing that traffic be classified according to the *service* it belongs to. A service is identified by a *serviceID*, which is the tuple {*server IP address, transport-level port ,protocol}*. Experimental data confirm that services are very stable even over long periods, making this method extremely simple, efficient and robust. Particularly, robustness is achieved because this method does not require the analysis of all sessions: provided that a service has been previously recognized, sessions accessing it can be classified even if encrypted at application-layer or data flow is observed only in one direction. Results in terms of efficiency are impressive, leading to a 37x reduction in processing cost, and a 20x reduction in the number of entries in data structures compared to session based classifiers at least in the traffic trace examined; furthermore each entry being half the size. Real-time measurements on the actual traffic transmitted on the upstream link of our University show that roughly 81% of the packets and 93% of the traffic (in terms of bytes) is successfully classified with the proposed method. Furthermore, service based classification is among the few methods that guarantee early classification, including the initial TCP handshake of a session. Among the few drawbacks of this method is the impossibility to classify IPsec traffic. It is worthy noticing that the precision of the service identification process is crucial for obtaining high-quality results, since a mismatch in service identification will impair the classification of all the sessions related to that service

REFERENCES

[1] Computer Networks Group (NetGroup) at Politecnico di Torino.The NetBee Library. August 2004.[online] Available at http://www.nbee.org/.

[2] S.Sen,O. Spatscheck,D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. Proceedings of World Wide Web Conference, pp. 512-521 NY, USA, May 2004.

[3] P. Haffner, S. Sen, O. Spatscheck, D. Wang, D. 2005. ACAS: automated construction of application signatures. In Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data, pp.197-202, Philadelphia, USA, August 2005.

[4] F. Risso, A. Baldini, M. Baldi, P. Monclus, O.Morandi. Lightweight, Session-Based Traffic Classification. Proceedings of the IEEE International Conference on Communications (ICC 2008) - Advances in Networks & Internet Symposium, Beijing, China, May 2008.

[5] F. Risso, A.Baldini, F. Bonomi. Extending the NetPDL Language to Support Traffic Classification. In Proceedings of IEEE Globecom 2007, Washington, D.C, USA, November 2007.

[6] R.Pang, V. Paxson, R. Sommer, L. Peterson.Binpac: a yacc for writing application protocol parsers. In Proceedings of the 6th ACM SIGCOMM on Internet Measurement, pages 289-300, Rio de Janeiro, Brazil, October 2006.

[7] O.Reviv. Inside network programming with SML. EE Times, August.

[8] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In Proceedings of ACM SIGCOMM, pages 229–240, Philadelphia, PA, August, 2005.

[9] A.Este, F. Gringoli,L. Salgarelli, Machine Learning techniques of traffic classification: an approach based on Support Vector Machines. Technical Report, November 2007.

[10] J. Erman, A. Mahanti, M. Arlitt. Traffic Classification using Clustering Algoritms. Proceedings ACM SIGCOMM Workshop on Mining Network Data (MineNet 06), Pisa, Italy, September 2006.

[11] J. Erman, A. Mahanti, M. Arlitt, C. Williamson. Identifying and Discriminating Between Web and Peer-to-Peer traffic in the Network Core. Proceedings of the 16th International World Wide Web Conference (WWW), pp. 883-892, Banff, Canada, May 2007.